

Experiment 2: TTL Digital Logic

Contents

1. Introduction	2
2. Introduction to Discrete Logic Circuits	2
3. TTL Integrated Circuits	4

1. Introduction

The purpose of this lab is to investigate the various logic circuits, from the discrete elements to integrated circuit chips. Generally you will use *Electronic Workbench (EWB)* software to simulate your logic circuits before actually constructing on the breadboard. For simple discrete circuits, choose any appropriate discrete components. To save time for discrete logic circuits, simulating on the *EWB* is sufficient to understand the operation. Instructions will be given at particular topics either to use *EWB* or to build the real circuit on the breadboard to compare with your simulated circuits.

2. Introduction to Discrete Logic Circuits

Logic circuits, in contrast to linear or analog circuits, are designed to carry two distinct signals: a high level or logic “one” and a “low” level or logic “zero” (See Figure 1). The choice of voltage levels is largely determined by the type of circuits used; unfortunately, as we will see later, it varies among several logic families that are commercially available. The assignment of logic “one” and “zero” states is arbitrary, but we will always follow the established positive logic convention – the more positive or “high” voltage represents a logic “one” state, the more negative or “low” voltage a logic “zero”.

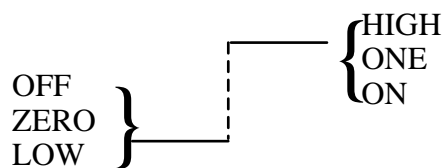


Figure 1. Logic level representation.

Use *EWB* to carry out the experiments from (A) through (D). Optimize the circuit parameters (such as resistance values) as necessary. Briefly describe how each circuit works.

A: Design (a) a two input diode *OR* gate and (b) a two input diode *AND* gate as shown in Figure 2. Choose suitable logic “zero” and “one” levels and define limits between the two states. Set up a truth table for (a) and (b) and verify it. List the advantages and disadvantages of simple diode logic.

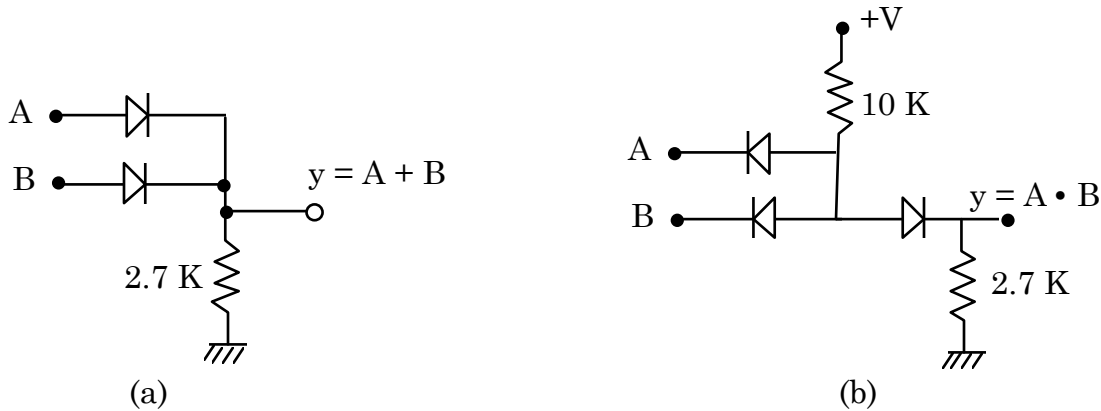


Figure 2. (a) 2-input diode *OR* gate. (b) 2-input diode *AND* gate.

B: Design the circuits 3(a) and 3(b), vary the input from 0 to +5 Volt, and plot the transfer function, v_o vs. v_i .

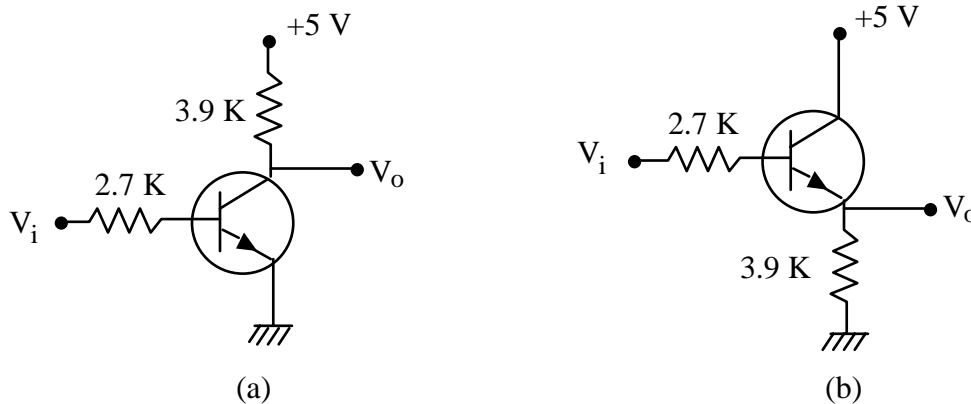


Figure 3. Transistor logic gates.

C: As an application of (3a) build a two input NOR gate. Observe the output voltage if one input is grounded and the second one is varied between 0 and +5 Volts.

D: The basic *DTL* (Diode–Transistor–Logic) *NAND* (Not AND) gate is shown below in Fig. 4. Briefly describe the behavior of this circuit.

- What are the output “0” and “1” levels?
- At what input voltage does the logic transition occur?
- Draw the V_o vs. V_i transfer characteristic.

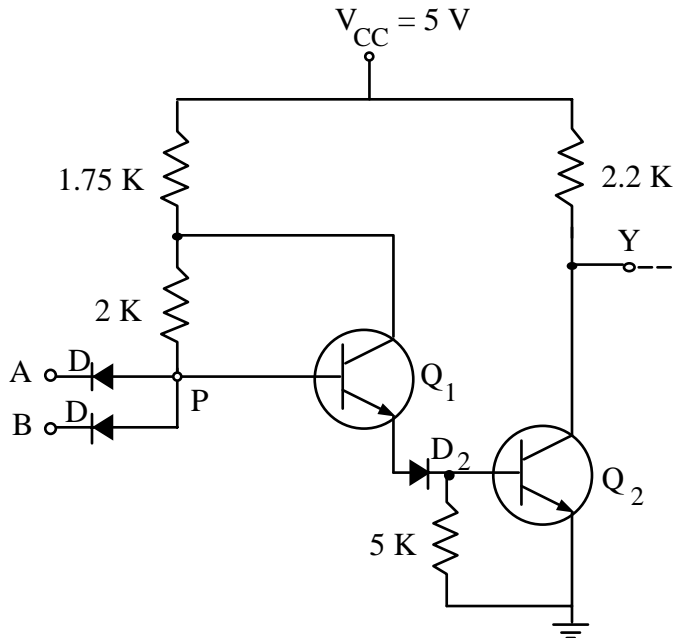
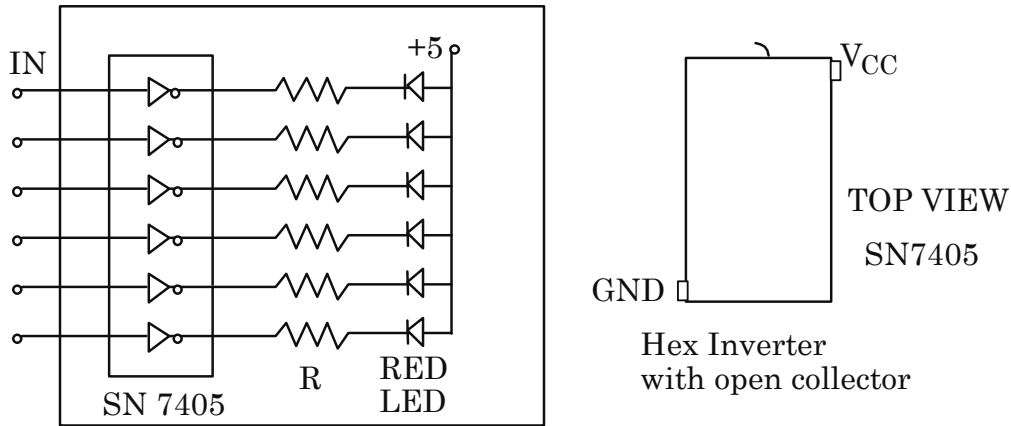


Figure 4. A basic Diode-Transistor-Logic (*DTL*) *NAND* gate.

3. TTL Integrated Circuits

In this part of this lab exercise, you will need to build selected circuits on the breadboard after simulating them with *EWB*. You will use integrated circuits of the *TTL* family (*SN 7400* Quad Two-Input *NAND*; *7402* Quad Two-Input *NOR*; *7404* Hex *Inverter*; *7473*, *7476* or *74107* Dual *JK Flip-Flop*; etc.) Look up maximum ratings, pin diagrams etc. in the Data Sheets folder available on your lab table, or in the *TTL* manuals, or from the data sheets available in Adobe PDF format on the P405 web page. Use the regulated +5 Volt supply mounted on each bench for V_{cc} and *ground*.

First, design an *LED* (Light Emitting Diode) indicator tester circuit as shown in Figure 5 and use it to monitor the voltage levels, *HIGH = Logic ONE*, *LOW = Logic ZERO*, at each point of your circuit. Save your test circuit for other parts of this lab.



LED Tester

Figure 5. LED tester and a typical pin layout of SN7405 IC. $R \approx 220 - 510 \Omega$.

A: Using *TTL NAND* and *NOR* Gates:

- a.) Construct an *AND* circuit on your breadboard using one portion of a *TTL Quad 7400 NAND Gate IC* and one portion of a *7404 Hex-Inverter IC*. Check (i.e.verify) the truth table.
- b.) Construct an *EXCLUSIVE OR (XOR)* function on breadboard using two *TTL Quad 7400 NAND Gate IC's* (n.b. not necessarily all 8 gates will be needed!) and check/verify the truth table.
- c.) Note that two logic symbols shown in Fig. 6 below are equivalent to each other. Prove it (i) by comparing the truth tables and (ii) by use of DeMorgan's theorem(s).

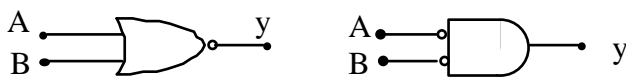
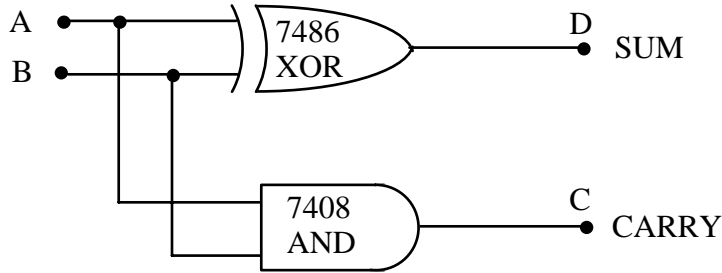


Fig. 6 Logical Equivalence of False NAND and Positive NOR Gates.

Revised October, 2004.

B: The 1-Bit Binary Half-Adder Circuit:

a.) Design and construct a 1-bit *half-adder* (Fig. 7) using the *AND* (7408) and the *EXCLUSIVE OR* (7486). Establish the truth table and verify it.



HALF ADDER

Fig. 7 One-Bit Binary Half-Adder Circuit.

b.) Using *EWB*, design a 1-bit *full-adder* and verify its truth table.

(See P405 lecture notes (online), or e.g. Taub & Schilling, page 357).

c.) (Optional) Design a two 8-bit word adder either by (1) using the 7483A (a 4-bit full adder) on the circuit board or (2) using *EWB* (use the IC model *CD4008* it supports in this case). Verify the function table. Note the typical add time in the manufacturer’s specification for 7483A.

C: The Set–Reset (S-R) Type Flip–Flop:

a.) Build the S-R flip-flop circuit shown in Fig. 8 and determine its truth table. Briefly describe how it works. Note that *changes* in logic levels take a few nsec to propagate through a *NAND* gate. One of the four possible logic combinations of \overline{S} and \overline{R} implies memory action – which one?

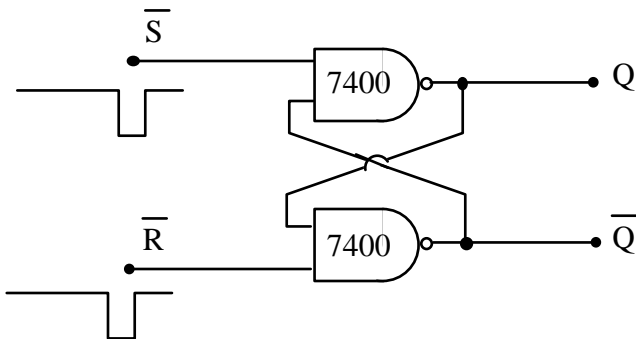


Fig. 8 The Set-Reset Type Flip-Flop.

Revised October, 2004.

b.) Mechanical Switch Contact Debouncing Circuit:

We will frequently need to use a circuit that enables us to *manually* produce clean logic “Zero” or “One” voltage levels, and also clean logic 0 → 1 (and logic 1 → 0) *transitions*. A mechanical switch, by itself, is *not* suitable for this purpose because of contact bounce – mechanical switch contacts *do not* close (or open) cleanly on short time scales! However, the use of an S-R type flip-flop *in conjunction* with the mechanical switch, as shown below in Fig. 9, known as a switch debouncing circuit, eliminates the problem associated with mechanical switch contact bounce!

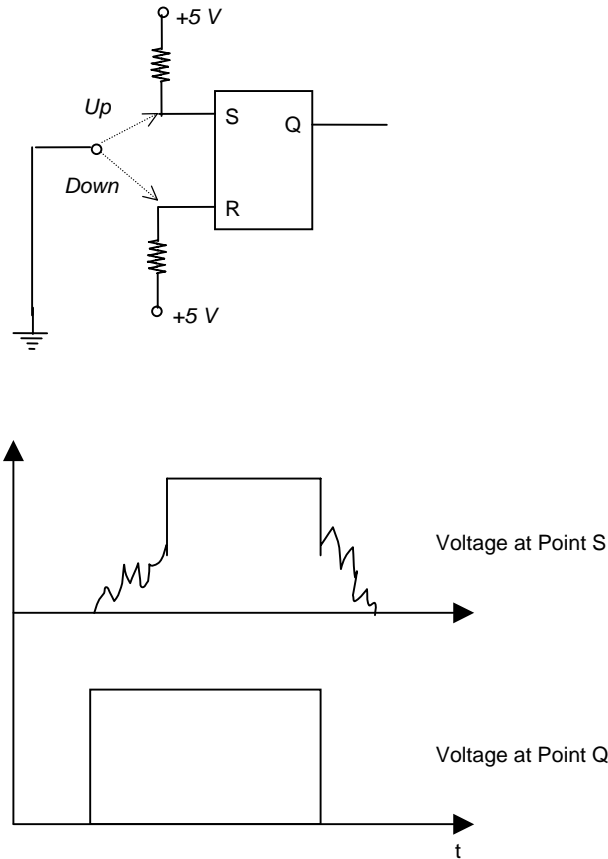


Fig. 9 Mechanical switch debouncing circuit using an S-R type flip-flop.

D: The D-Type Flip-Flop or Latch:

The S-R type flip-flop can be used as a one-bit memory device if we provide it with a suitable input circuit. To enter information, one of the two inputs should go low (see part C above), whereas both inputs should be maintained high to keep the information (latch). The circuits shown below in Fig. 10, called D-type flip-flops or “latches”, are two possible solutions. They have a data input, D and a clock input, Ck. Investigate the functioning of both circuits - you may use *EWB* alone for this section.

a.) Recognizing that the S-R flip-flop sub-circuits of the two D-type flip-flops shown below in Fig. 10 are identical – i.e. only the front-end portions of the two versions of the D-type flip-flops differ – work out the truth-table for both of the D-type flip-flop circuits shown in fig. 10 to verify their equivalence.

b.) What is the effect of $D = 1$ on the output, Q when $Ck = 0$ and $Ck = 1$ respectively? When does the information (data) enter the S-R flip-flop section of the D-type flip-flop? (n.b that Ck *transitions* from $0 \rightarrow 1$, and/or from $1 \rightarrow 0$).

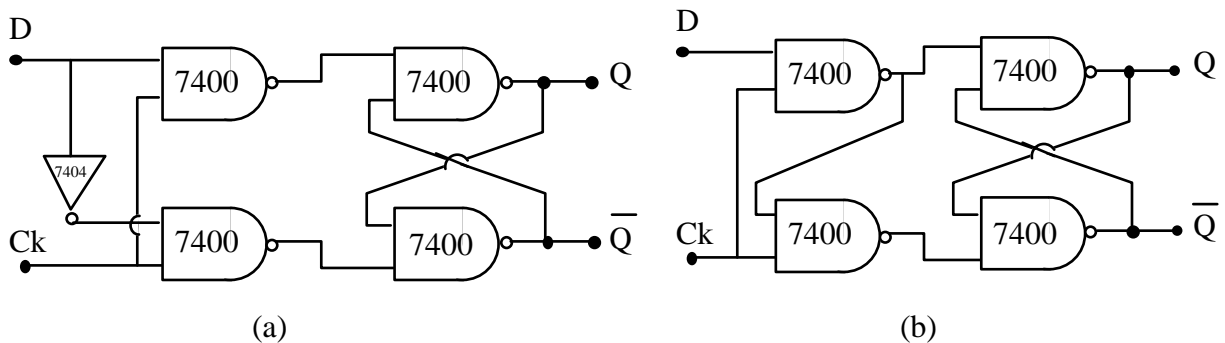


Fig.10 The D-Type Flip-Flop

E. The J-K Type Flip-Flop:

The J-K type flip-flop has *several* inputs, the J and K inputs, and also the Ck (Clock), Pr (Preset) and Cr (Clear) inputs as shown below in Fig. 11 for the TTL 7476 J-K type flip-flop.

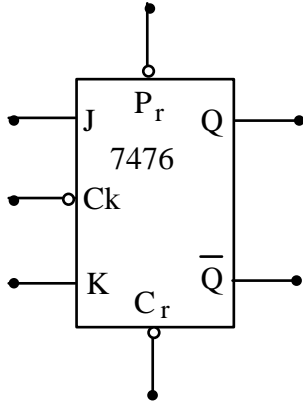


Fig. 11 The J-K Type Flip-Flop

a.) Referring to the datasheet for the TTL 7476 J-K flip-flop (see TTL Data Book; also available online on the P405 web page), explicitly compare the truth table for the 7476 J-K flip-flop with that for the S-R flip-flop – are there similarities between the two flip-flops? Build a circuit that will enable you to investigate the behavior of the 7476 J-K flip-flop on your breadboard. Use the switch contact debouncer circuit to provide Ck, and then explicitly verify/map out the entire truth table for the J-K flip-flop. Do your results agree with those on the 7476 datasheet?

b.) For $\{J = K = 1\}$ **.and.** $\{Pr = Cr = 1\}$, *toggle* is said to occur, i.e., each clock pulse on Ck reverses the state of the outputs, $Q \rightarrow \overline{Q} \rightarrow Q$. This mode is the basis of most binary counters. Determine at what point on the input Ck waveform, – i.e a positive-going (\lrcorner) or negative-going (\llcorner) edge/logic transition on Ck for which the toggling change of state in the output, Q occurs.

c.) With $\{J = K = 1\}$ **.and.** $\{Pr = 0, Cr = 1\}$ does the J-K flip-flop circuit *toggle* with each clock pulse on Ck?

F: The 4-Bit Parallel-Out Serial Shift Register:

- a.) Build a 4-bit parallel-out serial shift register circuit as shown below in Fig. 12 using 7476 J-K type flip-flops. Note that $K = \bar{J}$ for all flip-flops in this circuit. Wire up all four of the J-K flip-flop's Preset (Pr) and Clear (Cr) lines in such a way that the Q outputs (at points A, B, C & D) on all flip-flops can be simultaneously set to the $Q = 0$ (and/or all to the $Q = 1$) state. Explicitly verify this. Then, tie all Pr and Ck lines to logic high (+Vcc). Load a logic "1" into the first flip-flop (how?) so that $Q_A = 1$ and $Q_B = Q_C = Q_D = 0$, and then serially shift this bit (how?) through the shift register and explicitly verify this circuit's behavior (i.e. the state of the four Q outputs at points A, B, C & D) at *each* step in this shifting sequence. Would this circuit work with D-type flip-flops?
- b.) Compare this circuit with the functional block diagram of the TTL 74164 8-Bit Parallel-Out Serial Shift Register. Explain the added functions of the 74164.

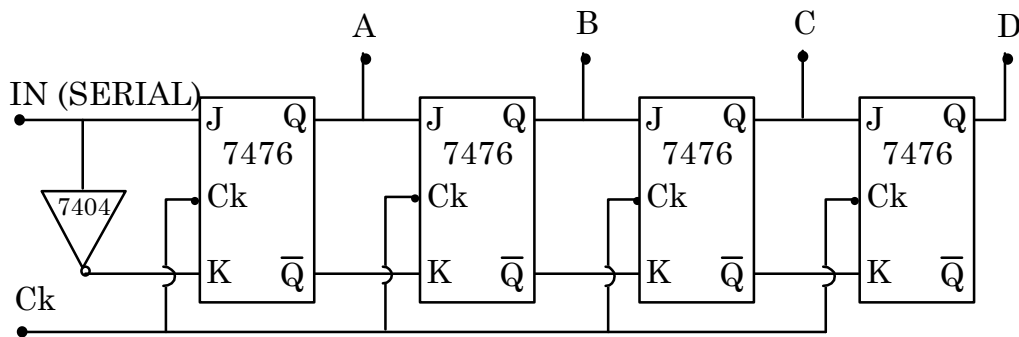


Fig. 12 4-Bit Parallel-Out Serial Shift Register Circuit using J-K Flip-Flops.

G: The Binary Ripple Counter:

- a.) Using EWB, design the circuit shown below in Fig. 13 and verify that it functions as a binary counter. Note that all J and K inputs should be held at logic “1”. Reset the counter with the C_r input. Would this circuit work with D-type flip-flops?

- b.) Compare this circuit with the functional block diagram of the TTL 7493A 4-Bit Binary Counter. Explain the added functions of the 7493A.

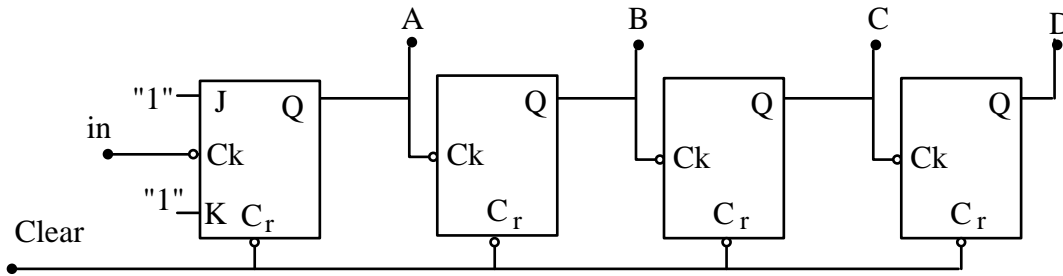


Fig. 13 Binary Ripple Counter Circuit using J-K Flip-Flops.

H. IC Oscillators:

- a.) Describe the nature of the oscillatory behavior of the circuit shown below in Fig. 14. What determines the oscillation frequency? What is your prediction for the oscillation frequency? Build this circuit and then measure the oscillation frequency. Is the measured oscillation frequency in reasonable agreement with your prediction?

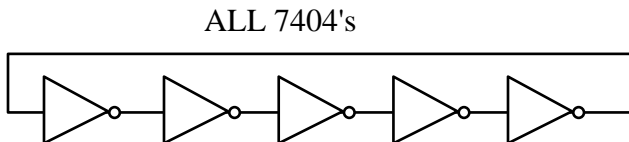


Fig. 14 IC-Based Oscillator Circuit Using 7404 IC's.

b.) Replace the 7404 Hex Inverter chip with a 74S04 Schottky TTL Hex Inverter. What is the new oscillation frequency? Does it differ from that obtained with previously with the use of the 7404 type chip? If so, explain the origin of the difference in oscillation frequencies.

c.) For a more stable oscillator, such as for use in a microprocessor as a clock, a quartz crystal can be used. Here again, use the (non-Schottky) SN7404 Hex Inverter IC. Build, and then measure the oscillation frequency of the circuit shown below in Fig. 15. Note that the third inverter is simply used as a buffer, in order to isolate the output from the actual clock circuit.

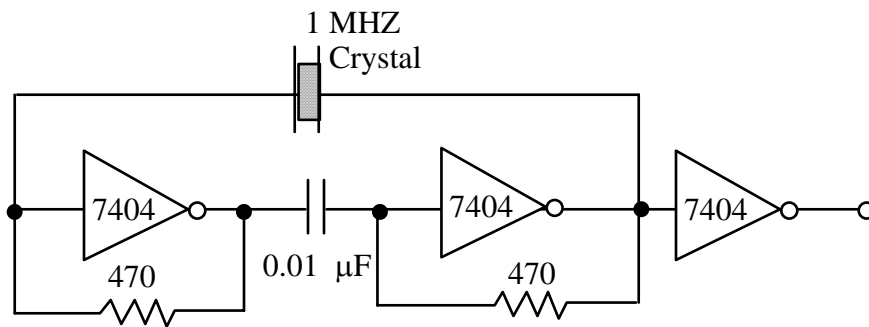


Fig. 15 IC-Based Oscillator Circuit Using 1 MHz Quartz Crystal.